EXHIBIT C

Marked Up Version of Amended Claim

1. (Amended) A system for storing, using and protecting access to [a master cryptographic key] sensitive data, comprising:

non-volatile storage;

a hidden storage location;

a first key derived from said sensitive data;

[a system initialization process that:

reads the master key from the non-volatile storage during a system initialization process;

writes a sensitive value derived from the master key to a hidden storage location;

disables access to the non-volatile storage by any program running in the system until the next start of system initialization process;]

means to prevent use of the first key [access to the hidden storage location] by programs running in the normal operating mode of the system; and,

means to allow use of the first key [access to the hidden storage location] by a program running in a restricted operating mode of the system.

2. (Amended) The system recited in Claim 1, wherein the sensitive data is the [master] first key.

3. (Amended) The system recited in Claim 1, wherein the sensitive data is derived [from the master key] using an application of a one-way cryptographic digest function of the first key.

4. (Amended) The system recited in Claim 3, wherein the sensitive data is a second key retrieved from encrypted data stored [on disk] in a storage medium, where the [stored] encrypted data is encrypted with [the master] a key derived from the first key.

5. (Amended) The system recited in Claim 1, wherein [software in BIOS ROM] firmware in a ROM or flash ROM controls the system during the system initialization process that begins in response to a power-on or reset signal.

6. (Amended) The system recited in Claim 1, wherein:

the non-volatile storage is non-volatile random access memory with read and write access controlled by a latch;

the latch is opened at the start of the system initialization process due to a hardware function responding to a power-on or reset event, thereby enabling system access to the non-volatile random access memory; and

the latch is closed during the system initialization process, thereby denying system access to the non-volatile random access memory until the next start of system initialization.

7. (Amended) The system recited in [Claims] Claim 1, wherein:

the hidden storage location is in a system management random access memory which is not accessible by any program running in the normal operating mode of the system; and

the restricted operating mode is a [System Management Mode] system management mode in which access to the system management random access memory is permitted.

8. (Amended) The system recited in [Claims] Claim 1, wherein:

the hidden storage is restricted for access by the operating system only, and is not accessible by any application program that runs in the normal operating mode of the system; and

the restricted operating mode is controlled by a CPU protection ring reserved for use by operating system software.

9. (Amended) A system for hiding a [master] cryptographic key in storage, comprising

power-on software that:

reads a [master] key from non-volatile storage;

closes access to the non-volatile storage such that access does not become available again until the next system reset;

writes sensitive data derived from the [master] cryptographic key to a hidden address space; and,

wherein only a program that runs in a restricted operational mode of the system has access to the sensitive data in the hidden address space.

14. (Amended) A method of controlling access to data by [to] an application program by restricting the use [availability] of a cryptographic key [to] for the application program on a [specific] device, comprising:

providing a first key known to a cryptographic processing module;

providing an application container data structure that contains a cryptographically sealed form of the data [that] for the application program [wants] to access;

performing a cryptographic gatekeeping function that [intercepts all access between application-level programs and the cryptographic processing module;

includes a means to examine a portion of the bytes of an executable in-memory image of a program that is attempting to access cryptographic services or data; and]

computes a cryptographic digest of a portion of [the bytes of] an in-memory image of the [calling] application program to compute [the] a cryptographic digest [transformation] of the application; and

performing an integrity-check [method performed] by the cryptographic processing module [that examines] by examining the application container data structure, the [and] cryptographic digest [transformation], and the [master] first key to determine if the application program is allowed to unseal the cryptographically sealed form of the data [in the given application container data structure, or when sealing the data modifies it to add the integrity check information].

15. (Amended) The method recited in Claim 14 further comprising performing a privacy [method performed] operation by the cryptographic processing module that encrypts or decrypts the cryptographically sealed form of the data in the application container data structure using a key derived from at least the [master] first key and cryptographic digest, [transformation] and when the cryptographically sealed form of the data is to be encrypted [it optionally], the crytographic processing module adds to the application container data structure the cryptographic digest [transformation] before the encryption is performed.

16. (Amended) The method recited in Claim 14 further comprising providing [the cryptographic gatekeeping function is concurrently or previously given] an authorization buffer that

specifies the result of the integrity-check [allowed operations for the application], and wherein [and] the cryptographic gatekeeping function confirms that the [request operation] application program is allowed to unseal the cryptographically sealed form of the data.

17. (Amended) The method recited in Claims 14 wherein the integrity-check [method] includes: [the steps of]

deriving a cryptographic variable from the cryptographic [transformation] digest and the [master] first key[, or of deriving a second cryptographic variable from the cryptographic transformation, the master key and a cryptographic variable chosen by a component of an application, and this derived key is used]; and

using the cryptographic variable to check a message authentication code that is stored in the application container data structure.

18. (Amended) The method recited in Claims 14 wherein the integrity-check [method] includes:

decrypting data derived from [a portion of] the application container data structure using a key derived from the first [master] key to create a resulting value and comparing [a portion of] the resulting value to data derived from [a portion of] the cryptographic digest [transformation,]; and

allowing [the] access to the cryptographically sealed form of the data if the [two portions are the same] resulting value is the same as the data derived from the cryptographic digest.

19. (Amended) The method recited in Claims [14] 15 wherein the privacy [step] operation includes: [the steps of]

deriving a cryptographic variable from the cryptographic [transformation] digest and the [master] first key [and optionally other information, or of deriving a second cryptographic variable

from the cryptographic transformation and the master key and a cryptographic variable chosen by a component of an application and optionally other information, and this derived key], wherein the cryptographic variable is used to decrypt or encrypt a portion of the application container data structure.

20. (Amended) The method recited in Claim 19 wherein the [key derivation] cryptographic variable is derived [is performed] with one or more applications of [the MD5 or SHA1 or SHA-256] a hash function[s] by concatenating [the] dependant values in [some] a particular order.

21. (Amended) The method recited in Claim[s] 14 wherein a portion of the cryptographic processing module executes during an system management interrupt.

22. (Amended) A method for authenticating an identified application program on an identified device to another computing machine comprising an authentication server with the help of another computing machine comprising a device authority, the method comprising:

[an enrollment process that includes the steps of:

a)] performing a first cryptographic enrollment operation [performed] during a system management interruption [(SMI)] on the identified device producing a result that is sent to the device authority; [, and

b)] performing a second cryptographic enrollment operation [performed] during [an SMI interrupt] the system management interruption on the identified device processing a value generated by the device authority that is received by the identified device;

[a registration process that includes the steps of:

a)] performing a first cryptographic registration operation [performed] during [an SMI] the system management interruption on the identified device [Device] producing a result that is sent to the authentication server[,];

[b)] performing a second cryptographic registration operation [performed] by the authentication server producing a cryptographic variable that is stored for use during the authentication method[, and];

[c) an optional third cryptographic operation performed during [an SMI interrupt] the system management interruption on the device processing a value generated by the authentication server that is received by the device;

an authentication process that includes the steps of:

a)] performing a first cryptographic authentication operation [performed] during [an SMI] the system management interruption on the identified device producing authentication data that is sent to the authentication server, and

[b)] performing a second authentication cryptographic operation [performed] by the authentication server on the authentication data received from the identified device using at least the cryptographic variable [stored during the registration method] to determine the result of the authentication.

23. (Amended) A method for authenticating an identified application program on an identified device, or for providing a second factor for identifying a user of the identified device to another computing machine comprising [a PASS] an authentication server, the method comprising:

[an application that

a) performs] performing an enrollment process including communicating [method involving communication] with a device authority and an authentication server to create an application

container data structure on the device, wherein the application container data structure is cryptographically associated with the application program; [and

b) stores] storing credential information, [and] wherein the authentication server stores a cryptographic variable for the application container data structure;

[an application running on the identified device that performs an authentication method including the steps of

a)] unsealing the application container data structure that stores the credentials;[,

b)] modifying the credentials;

[c)] resealing the application container data structure, wherein at least part of said resealing occurs during an SMI on the same CPU that executes the code of the application program;

[d)] sending identifying information and [at least a portion of] data derived from the resealed [AppContainer] application container data structure to the authentication server;

[wherein at least part of the resealing operation takes place during an SMI on the same CPU that executes the code of the application; and,

wherein the authentication server:

a)] receiving [receives] the identifying information and the [at least a portion of] data derived from the application container data structure; [,

b) uses] using the identifying information to lookup or compute a cryptographic variable to unseal the application container data structure; [,

c) if the unsealed application container has acceptable values then the specific application on a specific device is considered to be authenticated; and,

d) stores]

authenticating the identified application program and the identified device if the unsealed application container includes acceptable values; and

storing a key associated with the application container data structure.

24. (Amended) A [method] system for creating and utilizing one or more virtual tokens on a device for the purpose of at least one of authentication, privacy, integrity, authorization, auditing, [or] and digital rights management, each of said one or more virtual tokens having a corresponding type, the [method] system comprising:

an application program for each [kind of] of said corresponding type of virtual tokens;

an application container for each [virtual token of a specific kind;] of said corresponding type of virtual tokens; and

a cryptographic gatekeeping component that computes [an cryptographic transformation] a cryptographic digest of a calling application that is requesting cryptographic services of a cryptographic processing component;

[wherein the cryptographic gatekeeping component knows one or more long-lived symmetric keys;]

wherein the cryptographic processing component is accessed via the [CryptoGate] cryptographic gatekeeping component,[;]

wherein the cryptographic processing component knows a first key and a public key [one or more long-lived symmetric keys and one or more long-lived public keys; and],

wherein the cryptographic processing component performs cryptographic sealing and unsealing of application container data structures, where a portion of the cryptographic operations are performed during a system management interrupt [(SMI);],

wherein the cryptographic processing component checks the integrity of the calling application by checking a digital signature of a portion of the calling application's code or static

data, using [a] the public key [that has been loaded into] known to the [CryptoEngine] cryptographic processing component and a cryptographic digest [transformation] value[;],

wherein the cryptographic digest [transformation] value includes a recently computed cryptographic hash of a portion of the calling application's in-memory image[;], and

wherein the cryptographic gatekeeping and cryptographic processing component

a) derive a key for unsealing the application container data structure from the [master] first key and cryptographic digest [transformation],

b) use the derived key to check the message authentication code on the application container data structure, and returns an error if the message authentication code is correct, and

c) use the derived key to decrypt the data in the application container data structure and return it to the application.

25. (Amended) A method of securely associating a private key with an application program associated with a device, comprising:

creating an application container that contains private keys secured by a [symmetric] key associated with the application program and the device.